

New C Compilers: SDCC and AVR-GCC

Freeware for MCS51 and AVR!

By G. May, DL3ABQ

Alongside a wide variety of expensive, up-market products, two reasonably powerful C compilers for microcontrollers have emerged in the freeware circuit.

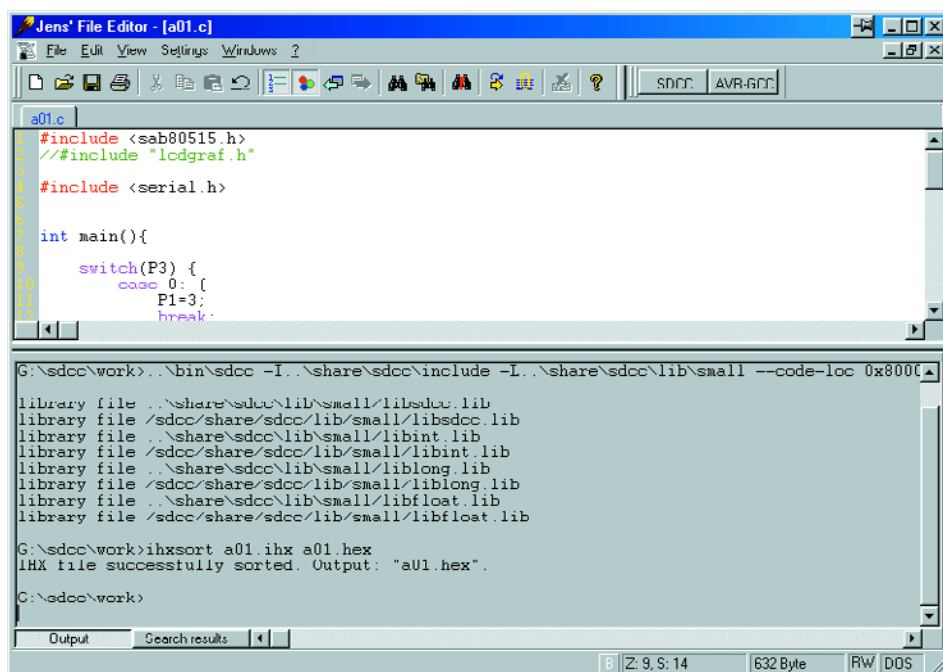


Figure 1. JFE in use.

The use of 'free' C compilers often presents unexpected problems because of the frequent lack of a user-friendly user interface. Also, despite their unrivalled price/performance ratio, these programs deter many potential users because their configuration and setup is not without problems. This article aims at providing some information to help overcome such initial problems. Specifically, we hope to supply some practical guidance in relation to the 'SDCC' compiler

for the MCS-51 series and 'AVR-GCC' for Atmel micros.

Editors

Because both compilers are in essence command-line driven programs, you will need an editor (word processing) utility to start with. This editor should allow programs to be written in a comfortable way. There

are some excellent, free, editors like JFE (Jens File Editor) and Proton, which offer roughly the same functionality.

Both editors support syntax highlighting (colouring of recognised 'C' terms) which is a great asset for spotting typos in your programs. They also allow external command line tools (like the C compilers discussed here) to be embedded into the user interface.

Such an editor enables you to write your own microcontroller program and compile it by calling the command line compiler. Once the compiler has finished its job, the resulting object code is ready for transferring (downloading) to the target microcontroller. The process of embedding the compilers into the editor is really simple and well described by the Online Help function.

Figure 1 shows JFE being used. As you can see, both SDCC and AVR-GCC have been embedded (buttons to the right of the menu). A program has just been compiled by SDCC and its output is shown by JFE in the lower area of the window.

Figure 2 shows the settings required for calling SDCC from JFE.

SDCC

SDCC (Small Device C Compiler) allows you to compile C programs for

devices from the 8051 series and a couple of other processors. The program is launched from the command line level by typing

```
sdcc <options> <file name>
```

Provided no errors are found, SDCC generates a file with the extension 'ihx', which contains the machine (object) code in IntelHex format.

Unfortunately, some monitor programs may run into problems because the lines in the IntelHex file have not been incrementally sorted by memory address. In order to overcome this difficulty, the author has written a small utility called `ihx-sort` that performs this sorting operation afterwards. The program may be obtained from the Free downloads section of the *Elektor Electronics* website under number **020339-11.zip**, which contains the source code as well as the executable (Windows) program. To prevent tedious typing over and over again it is recommended to write a batch file that calls the compiler with the relevant command line options. Of course, the parameters depend on the microcontroller system used (monitor program, memory configuration, etc.) as well as on the installation paths assigned to SDCC. Here

is a example of such a batch file:

```
(compile.bat):

c:\sdcc\bin\sdcc
-Ic:\sdcc\share\sdcc\include
-c:\sdcc\share\sdcc\lib\small
-code-loc 0x8000
-stack-loc 0x20 %1.c
ihxsort %1.ihx %1.hex
```

Calling the compiler requires this command line

```
compile <C source file without extension>
```

SDCC expects this batch file in the path `c:\sdcc`, compiles the program according to the `small` model for code position `8000h` (e.g., for a monitor program, with stand-alone applications the start address is usually `0000h`) and then arranges for the stack pointer to point to address `20h`. The resulting `ihx` file has the same name as the source file but the extension `.ihx`. At the same time, the program `ihxsort` is called and launched for the sorting work, producing an output file with the extension `.hex`.

The options available with the command lines for SDCC are numerous and a full overview is unfortu-

nately beyond the scope of this article. They may, however, be found in the user manual file `sdccman.pdf` contained in the previously mentioned zip file.

The speed of the generated code is not up to that of commercially available compilers like the industry standard Keil product. According to the manual, SDCC does some optimising during the compilation phase, but this is difficult to track or even prove in practice. None the less, SDCC is a good performer although it must be said that there is no substitute for assembly code in the case of time critical routines.

There is considerable variation in the quality of the library files supplied with SDCC, so the files should only be used after careful reading of the user manual mentioned above. Some example programs are supplied that allows the user to do a couple of quick tests to get started. Finally, there's a mailing list which you may join to discuss any problems that may occur.

AVR-GCC

AVR-GCC is the result of 'migrating' GCC (well known among UNIX fans) to Atmel AVR RISC controllers. The new compiler has seen rapid spreading across the Internet. Its machine code output is very powerful and comparable in performance with that produced by commercially available compilers.

With the Windows version, it is necessary to launch `run.bat` in the AVR-GCC subdirectory in order to set up a couple of important parameters for AVR-GCC. This happens automatically when the program is launched via the Start menu entry. Next, the user is free to go to the subdirectory containing his/her program and have it compiled.

AVR-GCC employs Make files for project management. The settings for compiling a particular project are stored in a file named `makefile`, while the compile process proper (in the relevant directory) is launched by calling

```
make
```

There are a number of standard 'Make' files available which only require small corrections to be able to use them for your own projects. During the installation, one such file is already stored in the subdirectory `avrgcc\avrfreaks`. For the purpose of your own projects, create a new directory, copy the `makefile 'template'` into it and adapt the paths, file names and any other options to suit your requirements.

During the compile process, AVR-GCC creates a `rom` and an `eep` file. The `rom` file con-

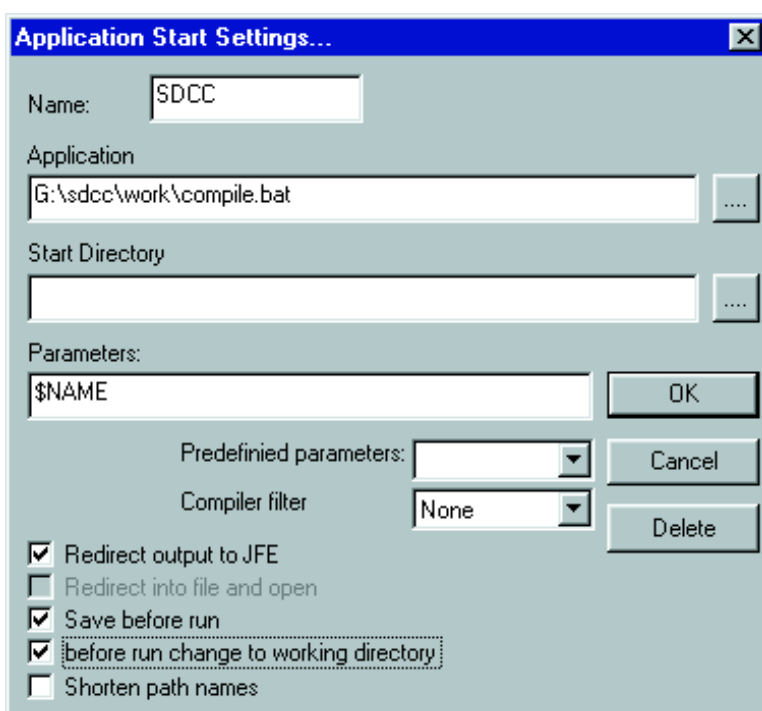


Figure 2. Settings to embed SDCC into JFE.

Internet addresses

SDCC download and mailing list

<http://sourceforge.net/projects/sdcc>

AVR-GCC download, examples and libraries

www.mikrocontroller.net/index.en.htm

JFE download

<http://home.t-online.de/home/Jens.Altmann/jfe.htm>

Proton editor

www.meybohm.de

tains the program for the microcontroller's Flash memory, and the eep file, the EEPROM contents — both are written in IntelHex format.

A number of excellent libraries are available for AVR-GCC and may be picked up from the Internet, just like example programs.

To close off this article, we'd like to mention a problem many AVR-GCC have reported in the AVR-GCC forum.

Just as many PC programs start by displaying a "Hello World" message, microcontroller applications

often indicate their initial activity by flashing an LED on a port pin. In many example ('demo') programs written for other microcontroller compilers, the time delay between the flashes is typically created by an empty for/next loop. Such constructs are, however, 'optimised away' by AVR-GCC. Consequently, time delays should be implemented using the controller's own timers. Alternatively, using volatile variables, the compiler should be prevented from removing such a loop.

(020339-1)